Jurnal Strategi
Volume 3 Nomor 2 November 2021

Implementasi *Automation* dalam Perbandingan Data pada *Website* Pokemon

Alvira Puteri[#1], Setia Budi, Ph.D[*2]

[#]*Program Studi Sistem Informasi, Universitas Kristen Maranatha*
Jl. Surya Sumantri No.65 Bandung 40164
[1]1873004@maranatha.ac.id
[2]setia.budi@it.maranatha.edu

Abstract — ***Around 2016, the public was enlivened with Pokemon characters in the PokemonGo game. The Pokemon character, which first appeared in 1995, managed to attract people's attention and become a topic of conversation around the world. Until now, information about Pokemon continues to grow along with the increasing number of Pokemon that continue to be found. This is certainly a difficulty for the public to be able to recognize and remember all information related to Pokemon. Over time, websites containing the Pokedex, an electronic encyclopedia that stores information on all registered Pokemon, began to appear. Three of these websites are pokemondb.net, bulbapedia.bulbagarden.net and pokeapi.co. Unfortunately, some websites are created just to follow trends and do not necessarily store accurate information. One way to ensure the correctness of the information in it is to compare the information from one website to another. Given the limitations of humans in comparing all data manually, by copying the data and comparing the data one by one, then another way that can be done is to compare the data automatically using automation.***

Keywords— ***Automation, Comparison, Information, Pokemon.***

I. PENDAHULUAN

Setiap harinya, kondisi dunia pekerjaan terus berubah seiring dengan perubahan kebutuhan perusahaan. Tingginya tingkat lulusan di bidang teknologi informasi yang disertai dengan minimnya pengalaman menjadi salah satu kekhawatiran perusahaan saat ini, terutama BliBli. Berangkat dari hal tersebut, BliBli kemudian mengadakan sebuah program pelatihan bernama *BliBli Future* yang bertujuan untuk mempersiapkan para mahasiswa dengan memberikan materi serta studi kasus yang nyata digunakan dalam perusahaan Blibli. Selama program berjalan, mahasiswa akan menerima pelatihan, dan ditutup dengan pemberian project yaitu pembuatan *automation* dengan studi kasus Pokemon.

Pokemon / *Pocket Monster* merupakan spesies monster fiksi yang pertama kali muncul dalam permainan video yang diciptakan oleh Satoshi Tajiri pada tahun 1995 [1]. Nama *Pocket Monsters* sendiri merujuk pada sebuah aktivitas ikonik pada seluruh serial Pokemon, di mana spesies monster Pokemon dapat dimasukkan ke dalam Pokeball yang disimpan dalam pocket / saku. Tingginya minat masyarakat di seluruh dunia terhadap karakter Pokemon [2], kemudian menjadikan karakter Pokemon diadaptasikan ke dalam berbagai bentuk, seperti film, buku, trading cards, dan lainnya.

Dalam permainan video, pemain disebut sebagai *pokemon trainer* dan memiliki misi utama yaitu mengoleksi dan melengkapi isi dari Pokedex, dengan cara mencari seluruh spesies Pokemon yang terdaftar dalam Pokedex. Pokedex sendiri merupakan sebuah ensiklopedia elektronik dalam permainan yang menyediakan berbagai informasi tentang Pokemon. Para *pokemon trainer* dapat mengumpulkan karakter Pokemon dengan cara menjelajahi dunia nyata di sekitarnya,dan berburu Pokemon liar [3] . Setiap Pokemon yang ditemui dapat dijadikan sebagai peliharaan dengan melemparkan *pokeball* yang dapat menciutkan massa jenis Pokemon.

Selain mengumpulkan spesies Pokemon, *Pokemon Trainer* juga dapat mengejar gelar *Pokemon Master*, yang merupakan *Pokemon Trainer* terkuat [4]. Untuk menjadi *Pokemon Master, Pokemon Trainer* perlu masuk ke kejuaraan nasional Liga Pokemon, mengalahkan empat sesepuh Pokemon atau yang lebih dikenal sebagai *The Elite Four*, dan mengalahkan Juara Regional.

Hingga saat ini terdapat sekitar 890 spesies Pokemon yang telah dikonfirmasi dan tidak menutup kemungkinan untuk terus bertambah di masa yang akan datang. Mengingat jumlah Pokemon yang besar, tidak semua spesies Pokemon dapat dikenal dan diingat. Seiring berjalannya waktu, berbagai *website* yang berisi informasi seputar Pokemon mulai bermunculan, tiga di antaranya adalah pokemondb.net, bulbapedia.bulbagarden.net dan pokeapi.co. Sayangnya, tidak semua *website* menyajikan

informasi yang akurat dan hal tersebut mendasari penelitian ini di mana penelitian ini dilakukan untuk membandingkan informasi dari ketiga *website* tersebut dengan berfokus pada pembuatan *automation comparison*.

II. KAJIAN TEORI

A. Automation

Automation / test automation merupakan salah satu cara melakukan pengujian dengan bergantung pada *tool* atau *scripts* yang berjalan secara otomatis melalui penggunaan framework dan dirancang untuk membandingkan hasil yang diharapkan dengan hasil yang sebenarnya sehingga menghasilkan laporan yang menyatakan suatu tes berhasil atau gagal [5]. Melalui penggunaan *test automation,* seorang tester dapat melakukan beberapa pengujian sekaligus sehingga dapat membantu tester dalam membandingkan data dengan waktu yang relatif singkat dan dapat meningkatkan efektifitas dan efisiensi pengujian [6].

B. Data Scraping

Kegiatan perbandingan data sangat bergantung pada ketersediaan data yang akan diuji sehingga aktivitas ini perlu diawali dengan melakukan proses pengambilan data dari *website* yang sering disebut dengan istilah *data scraping* [7]. *Data scraping* mampu meningkatkan efisiensi proses analisis data dengan membantu mengumpulkan seluruh data yang diperlukan secara lengkap, baik data dari log tidak terstruktur atau log semi terstruktur, kemudian menyimpannya dalam penyimpanan data yang diinginkan [8]. Proses *data scraping* dapat dilakukan melalui dua cara, yaitu dilakukan secara manual, dengan melakukan *copy paste* data satu persatu, atau dilakukan secara otomatis, dengan melakukan ekstraksi isi *website* menggunakan teknik [9] salah satunya teknik Xpath [10]. Xpath, atau yang lebih dikenal dengan *XML Path Language*, merupakan bahasa *query* yang bekerja dalam dokumen XML dengan menavigasi struktur dokumen melalui pemilihan *nodes* dari struktur file XML dan HTML [11]. XPath menggunakan ekspresi untuk menghubungkan *node* konteks dengan *node* yang sedang dicari sehingga dapat menyaring *nodes* yang ada untuk mengekstrak konten.

C. Selenium

Selenium adalah salah satu *tool automation testin*g yang bersifat *open source* dan berfokus pada otomatisasi tes pada aplikasi web di berbagai *browser* [12]. Selenium sendiri terdiri dari serangkaian perangkat lunak, salah satunya *webdriver*. *Webdriver* ini nantinya akan berperan sebagai medium untuk mengontrol pengujian interaktif pada *website,* melalui otomatisasi dengan bahasa pemrograman [13]. Melalui Selenium, klien dapat mengirimkan *script* pengujian melalui *driver* khusus *browser*. *Driver* tersebut kemudian akan menjalankan *script* pengujian pada *instanc*e *browser* seperti untuk membuka url, mencari elemen dll.

D. TestNG

TestNG, atau *Test Next Generation*, adalah *framework testing* yang populer dikalangan tester dimana TestNG diciptakan dengan terinspirasi dari *framework* lain dan bertujuan untuk mengatasi masalah ataupun kekurangan pada *framework* tersebut. Oleh karena itu, TestNG memiliki beberapa fitur tambahan yang tidak dimiliki *framework* lain dan membuatnya lebih unggul. TestNG juga bersifat *open source* dengan memiliki lisensi dari *Apache Software Foundation* [14]. Dengan menggunakan TestNG , tester dapat melakukan konfigurasi pengujian dengan lebih mudah dan menghasilkan laporan pengujian yang layak dimana TestNG didesain untuk melakukan *end-to-end testing* [15].

III. ANALISIS DAN RANCANGAN SISTEM

A. Analisis Sistem

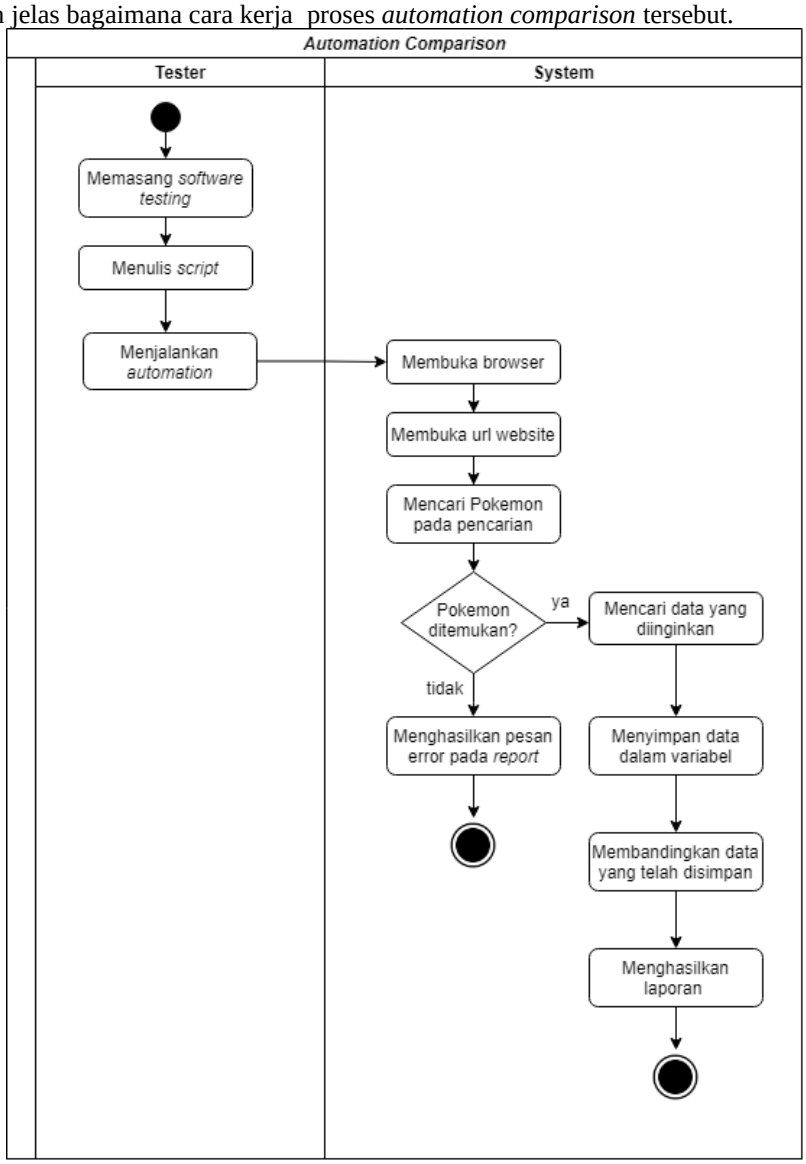
Setidaknya terdapat tiga *website* Pokemon, *website* berisi informasi seputar Pokemon untuk dijadikan studi kasus dalam penelitian ini, yaitu pokemondb.net, bulbapedia.bulbagarden.net dan pokeapi.co. Ketiga website ini juga sudah dikenal luas di kalangan pecinta Pokemon. Adanya kemungkinan akan *website* yang tidak dibuat dengan informasi yang benar melainkan mengambil sembarang data hanya untuk mengikuti tren, mendasari penelitian ini untuk melakukan analisis perbandingan performa informasi terhadap ketiga *website* tersebut dengan tujuan memastikan kesamaan data dalam ketiga sumber data. Dalam penelitian ini, pengujian akan berfokus pada *automation comparison* dengan menggunakan salah satu *tool automation* yang bersifat *open-source*, yaitu *Selenium*, mengingat pengujian secara manual akan memakan waktu yang cukup lama dan tidak efektif.

Dengan *automation*, tester hanya perlu menjalankan *script* yang telah disusun melalui bahasa pemrograman, dalam hal ini Java, dengan menggunakan bantuan Selenium *webdriver*. *Webdriver* tersebut nantinya akan membuka web browser dan

mengeksekusi *script* yang dijalankan termasuk salah satunya untuk membuka *url*. Setelah *website* yang dituju terbuka, teknik *data scraping* akan berperan dalam pengambilan data / informasi yang akan dibandingkan dan menyimpannya ke dalam variabel. Proses perbandingan data kemudian akan menghasilkan sebuah *report* yang menyatakan apakah data sama atau terdapat perbedaan.

B. Rancangan Sistem

Gambar 1 merupakan gambaran proses *automation comparison* yang saat ini berjalan. Gambaran ini bertujuan untuk menunjukkan secara lebih jelas bagaimana cara kerja proses *automation comparison* tersebut.



Gambar 1. *Flowchart* Sistem Berjalan

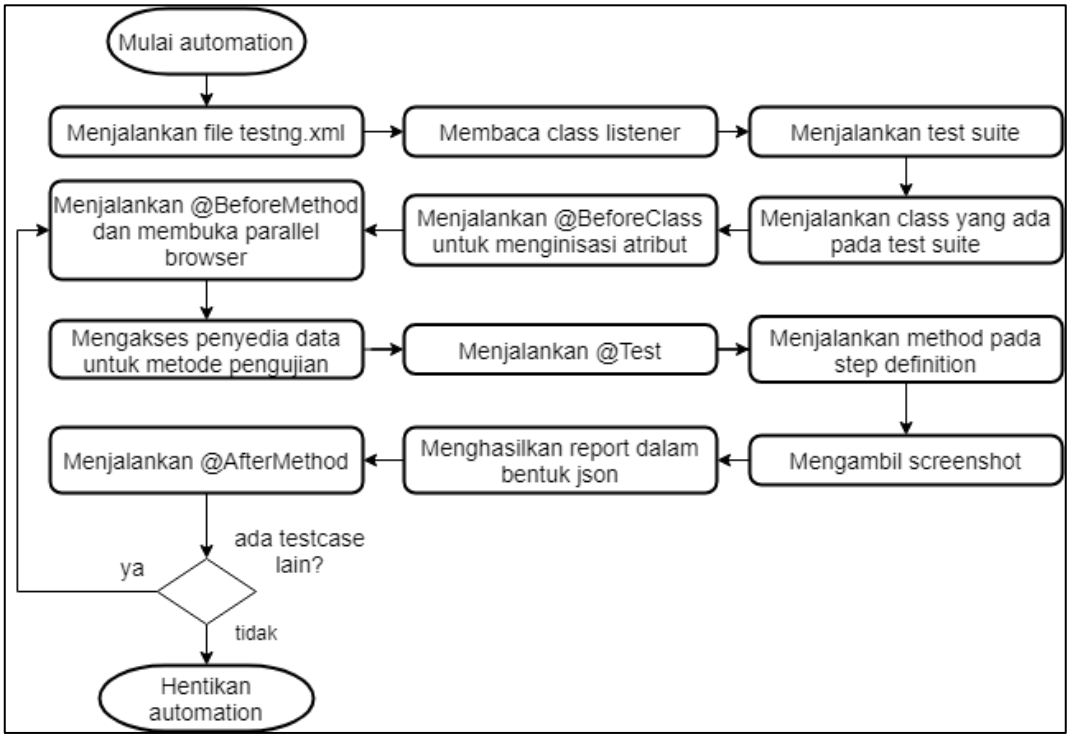
Dalam penelitian ini, untuk melakukan *automation comparison*, tester akan dibantu oleh *software testing tool,* yaitu Selenium, yang dapat dipasang dengan menambahkan *dependency* pada file pom yang ada dalam *project*. Setelah memastikan *tool* tersebut dapat berjalan dengan baik, tester dapat langsung menuliskan *script* sesuai dengan kebutuhan pengujian. Aktivitas *automation comparison* bergantung pada penulisan *script* yang dituliskan pada *software testing*, oleh karena itu penulisan *script* merupakan tahapan proses yang sangat penting dalam *automation comparison*. *Script* yang telah dituliskan, kemudian akan dijalankan oleh tester dengan bantuan *webdriver*. *Webdriver* akan bertugas untuk mengontrol *browser* dan melakukan eksekusi *script* pada *instance browser* yang diminta.

Sesudah *webdriver* telah berhasil membuka *instance browser* yang diminta, *webdriver* akan membuka alamat *website* yang ada pada *script*, kemudian menjalankan teknik *data scraping* melalui Xpath yang juga telah didefinisikan dalam *script*. Xpath akan memilih node dari struktur XML atau HTML yang diinginkan, namun Xpath berkemungkinan untuk tidak menemukan data yang diinginkan jika terdapat kesalahan dalam penulisan Xpath atau jika tester memasukkan *test case* negatif untuk pengujian. Jika data yang diinginkan tidak dapat ditemukan, maka sistem akan menambahkan pesan *error* pada *report* dan jika Xpath berhasil menemukan data yang diinginkan, data tersebut akan tersimpan dalam sebuah variabel yang telah ditetapkan dalam *script* untuk nantinya dibandingkan dengan data dari *website* lain.

Hasil perbandingan dari ketiga *website* tersebut, nantinya akan tersimpan dalam sebuah *report*. *Report* ini akan dapat menyimpan pesan error dan *screenshoot* UI (untuk *website* UI) jika terdapat pengujian yang gagal, dan akan menyimpan pesan berhasil jika data berhasil ditemukan dan perbandingan data yang dilakukan memiliki informasi yang sama.

IV. IMPLEMENTASI

Setelah sistem dianalisis dan dirancang secara terperinci, tahap berikutnya adalah implementasi. Adapun pelaksanaan atau penerapan dari perancangan sistem tersebut dijabarkan dalam gambar 2.



Gambar 2. Flowchart Implementasi Automation Comparison

Pada saat *automation* dijalankan, sistem akan membaca file testng.xml yang merupakan file pusat konfigurasi dari seluruh pengujian yang dijalankan menggunakan *framework* testNG. Pada file xml tersebut terdapat beberapa konfigurasi, salah satunya adalah konfigurasi untuk *listener*. Ketika file xml dijalankan, sistem akan membaca konfigurasi *listener* tersebut dan mencari *package* berisi class listener yang mengatur perilaku testNG, seperti menambahkan *screenshot* pada pengujian pada *website* UI maupun menyertakan API *request-response* pada pengujian API.

Tidak hanya mengatur konfigurasi terkait *listener*, file testng.xml juga memiliki konfigurasi untuk setiap *test suite* yang akan dijalankan. Konfigurasi tersebut meliputi nama *test suite*, parameter pengujian, jenis paralel pengujian, serta *class runner* yang akan dijalankan untuk setiap *test suite*. Setelah file xml membaca *class listener*, selanjutnya sistem akan menjalankan *test suite* dan *class runner* sesuai dengan konfigurasi yang ada. Pada setiap *class runner* yang dijalankan, sistem akan melaksanakan pengujian berdasarkan jenis anotasi yang digunakan.

Setidaknya terdapat empat anotasi utama yang digunakan dalam *class runner*, di antaranya *@beforeclass, @beforemethod, @test,* dan *@aftermethod.* Masing-masing anotasi tersebut memiliki fungsi yang berbeda. Anotasi

@beforeclass mengindikasikan bahwa metode dengan anotasi tersebut akan dijalankan sebelum metode pertama di *class* tersebut dijalankan. Sedangkan anotasi *@beforemethod* mengindikasikan bahwa metode dengan anotasi tersebut akan dijalankan sebelum setiap metode *@test* dijalankan. Anotasi *@test* sendiri mengindikasikan bahwa method yang memiliki anotasi tersebut merupakan metode berisi logika utama dalam pengujian dan akan dijalankan ketikan metode dengan anotasi *@beforemethod* selesai dieksekusi. Biasanya setiap metode pengujian akan diakhiri dengan metode beranotasi *@aftertmethod* dimana metode tersebut akan dijalankan saat setiap metode *@test* selesai dieksekusi.

Sesuai dengan urutan anotasi yang mengiringinya, metode dengan anotasi *@beforeclass* akan dijalankan setelah sistem menjalankan sebuah *class runner*. Metode ini berfungsi untuk menginisiasi seluruh variabel yang akan digunakan dalam tes. Setelah itu dilanjutkan dengan menjalankan metode dengan anotasi *@beforemethod* yang berisi inisiasi *webdriver*. Berdasarkan konfigurasi pada file xml, setiap tes yang ada pada *class runner* akan dijalankan secara paralel, dalam hal ini adalah paralel *browser*, sehingga pada saat dijalankan, *webdriver* akan membuka 2 *browser* sekaligus, yaitu *chrome* dan *firefox*. Kemudian, sistem akan mengakses metode penyedia data atau *data provider* yang menyimpan parameter Pokemon untuk setiap metode pengujian dan dilanjutkan dengan menjalankan metode pengujian tersebut.

Ketika metode dengan anotasi *@test* dieksekusi, sistem akan mengakses *class* yang berisi *step definition* dan menjalankan semua metode yang ada di dalamnya dengan parameter yang diberikan oleh metode penyedia data. Pada metode pengujian ini, *webdriver* akan membuka *url* dari ketiga *website* yang menjadi studi kasus, kemudian melakukan pencarian berdasarkan parameter Pokemon yang ada. Ketika parameter yang diinginkan telah ditemukan, sistem akan mencari data yang diminta dan menyimpannya dalam variabel. Setiap data yang didapatkan akan dibandingkan antara satu *website* dengan *website* lain. Saat tes selesai, sistem akan menyimpan hasil pengujian, baik pengujian tersebut berhasil maupun gagal, dalam bentuk json lalu mengambil *screenshot* serta API *request - response* dan menyimpannya sebagai attachment.

Selanjutnya, sistem akan menjalankan metode yang memiliki anotasi *@aftermethod* dan menutup *webdriver* yang sedang berjalan. Proses ini dilanjutkan dengan melakukan pengecekan untuk memastikan apakah seluruh metode pengujian telah dijalankan. Jika masih terdapat metode dengan anotasi *@test* yang belum dijalankan, sistem akan mengulangi alur pengujian hingga seluruh tes selesai dijalankan. Apabila sudah tidak ada metode beranotasi *@test* yang perlu dijalankan, maka sistem akan menghentikan *automation.*

V. PENGUJIAN

A. Rencana Pengujian

Sebelum implementasi pembuatan *automation comparison* ini dikatakan layak, perlu dilakukan suatu pengujian untuk memastikan dan menunjukkan bahwa sistem yang dikembangkan telah sesuai dengan tujuan pengembangan serta kebutuhan pengguna. Selain itu, tahap pengujian juga dilakukan untuk mencari dan menemukan kesalahan yang mungkin muncul selama pengembangan sistem. Dalam penelitian ini, pengujian yang digunakan adalah pengujian dengan metode *black box* testing, di mana pengujian terhadap *automation* ini dilakukan dengan menggunakan beberapa *test case* untuk setiap metode pengujian yang terdapat dalam sistem. Metode testing ini nantinya akan memperlihatkan hasil keluaran yang diharapkan dari sistem terhadap *test case* yang dijalankan, hasil keluaran yang sebenarnya dihasilkan oleh sistem terhadap *test case* yang dijalankan, dan hasil pengujian berupa kesimpulan

B. Pengujian Black Box Testing

Berikut merupakan hasil pengujian *black box testing* terhadap *automation comparison* untuk *website* Pokemon.

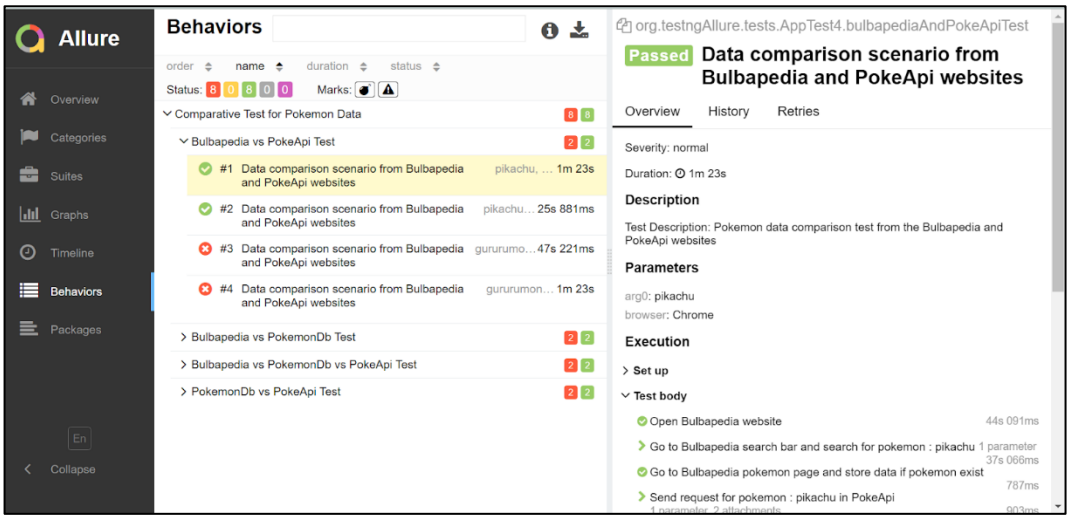
TABEL 1
PENGUJIAN TERHADAP AUTOMATION COMPARISON

No	*Test Case*	Hasil Harapan	Hasil Keluaran	Hasil Uji
1.	Positive (Memasukkan parameter Pokemon yang valid)	Pengujian berhasil dan tidak menampilkan pesan error serta menampilkan screenshot untuk UI dan API request-response untuk API	Pengujian berhasil dan tidak menampilkan pesan error serta menampilkan screenshot untuk UI dan API request-response untuk API	*Valid*
2.	Negative (Memasukkan parameter Pokemon yang tidak valid)	Pengujian gagal dan menampilkan pesan error yang menunjukkan alasan pengujian gagal serta menampilkan	Pengujian gagal dan menampilkan pesan error yang menunjukkan alasan pengujian gagal serta menampilkan	*Valid*

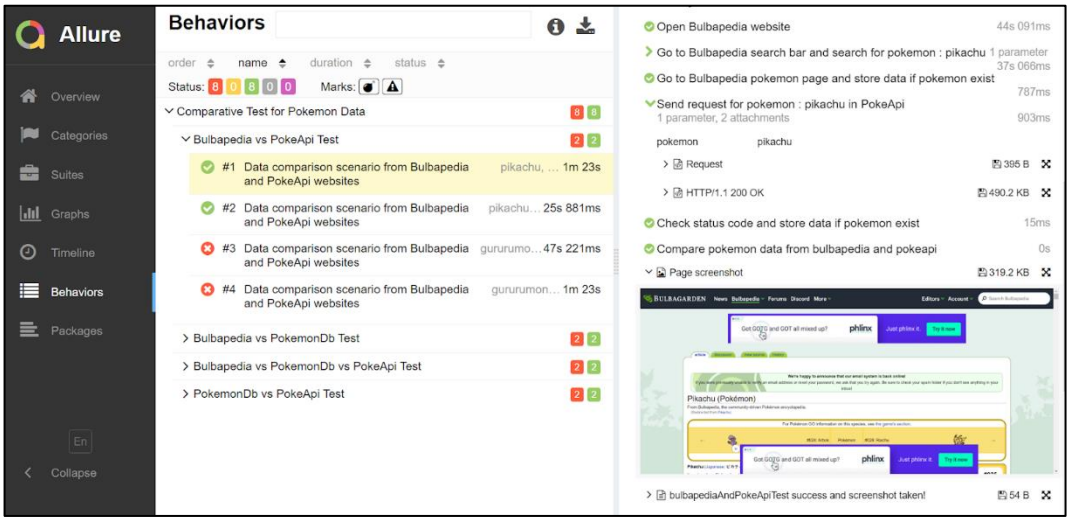
		screenshot untuk UI dan API request-response untuk API	screenshot untuk UI dan API request-response untuk API	

C. Hasil Pengujian

Dalam penelitian ini, ketika tahap implementasi telah selesai dilakukan, sistem akan menghasilkan sebuah laporan dalam bentuk *json* dan *attachment* yang tersimpan dalam sebuah folder. *File-file* dalam folder tersebut nantinya dapat dikonversi kedalam bentuk *HTML* dengan menggunakan *command line allure serve*. Ketika *command line* tersebut dijalankan, tampilan HTML untuk laporan yang tersimpan dalam folder tersebut akan terbuka dalam *web browser*, dalam hal ini *Chrome*, dan akan memiliki tampilan seperti gambar 3,4 dan 5.

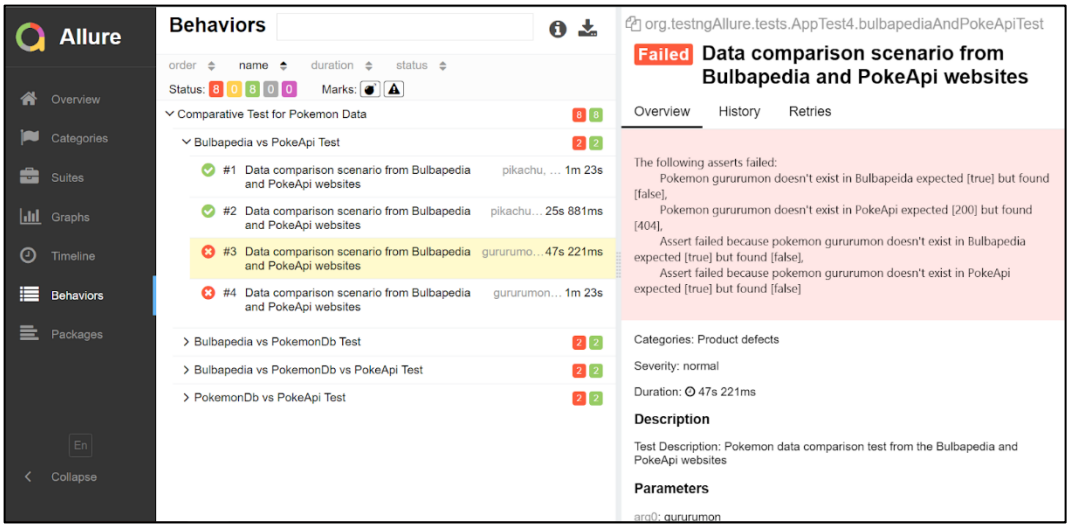


Gambar 3. Laporan dengan *test case* positif (1)



Gambar 4. Laporan dengan *test case* positif (2)

Dari tampilan HTML pada Gambar 3 dan 4, sebuah pengujian yang berhasil akan diindikasikan dengan simbol berwarna hijau dan untuk pengujian yang gagal akan diindikasikan dengan simbol berwarna merah. Jika hasil pengujian berhasil (dengan kata lain data-data yang dibandingkan sudah sama), maka laporan akan menampilkan informasi berupa nama metode, durasi eksekusi, deskripsi, parameter pengujian, isi dari langkah-langkah pengujian, API *request-response* (jika pengujian dilakukan pada *website* API), dan *screenshot* (jika pengujian dilakukan pada *website* UI).



Gambar 5. Laporan dengan *test case* negative

Di sisi lain, pada Gambar 5, jika hasil pengujian gagal (dengan kata lain terdapat data atau beberapa data yang memiliki perbedaan antar *website*), maka informasi yang sama akan ditampilkan, namun dengan menyertakan informasi tambahan, yaitu informasi terkait error yang didapatkan dari metode yang bersangkutan. Ketersediaan informasi tersebut berguna untuk mempermudah pengguna dalam menelusuri hasil pengujian. Informasi terkait setiap kesalahan yang muncul dalam tahapan yang dijalankan akan turut disertakan pada laporan.

VI. SIMPULAN DAN SARAN

A. *Simpulan*

Berdasarkan penelitian yang telah dilakukan oleh peneliti serta pembahasan pada bab-bab sebelumnya, maka kesimpulan yang dapat diambil berdasarkan tujuan dari penelitian ini adalah sebagai berikut.

1. Implementasi *automation comparison* pada studi kasus w*ebsite* dan API Pokemon dapat dilakukan untuk mengoptimalkan pemrosesan data dan menghasilkan data analisis perbandingan yang diinginkan dengan menggunakan *tool automation* dan *framework testing* serta melalui tahapan implementasi pada pembahasan bab-bab sebelumnya
2. Perbandingan kesamaan data pada tiga sumber data (dua *website* Pokemon dan satu API Pokemon) sangat memungkinkan untuk dilakukan dan dapat menjawab kebutuhan pengguna dalam hal perbandingan data.

B. *Saran*

Automation comparison dengan konsep perbandingan data seperti pada studi kasus penelitian ini dapat dikembangkan dengan cakupan yang lebih luas dan juga dapat diimplementasikan dalam kasus lain untuk membantu pengguna dalam mempermudah proses membandingkan data pada suatu *website* maupun API. Penggunaan *automation comparison* juga dapat digunakan untuk membantu dan mempermudah pengguna dalam mengidentifikasi validitas dari informasi yang disertakan oleh sejumlah *website*.

DAFTAR PUSTAKA

[1] V. Suryaputra, "ilmupedia," 29 June 2015. [Online]. Available: https://ilmupedia.co.id/articles/pokemon-sejarah-dan-generasinya-hingga-kini/full. [Diakses 9 March 2021].
[2] A. Malik, K. Hiekkanen, Z. Hussain, J. Hamari dan A. Johri, "How players across gender and age experience Pokémon Go?," *Springer*, 2019.
[3] L. D. Kaczmarek, M. Misiak, M. Behnke, M. Dziekan dan P. Guzik, "The Pikachu Effect: Social and Health Gaming Motivations Lead to Greater Benefits of Pokémon GO Use," *Computers in Human Behavior*, 2017.
[4] K. F. &. T. R. J. Mitchell Vaterlaus, ""Reliving my Childhood Dream of being a Pokémon Trainer": An Exploratory Study of College Student Uses and Gratifications Related to Pokémon Go," *International Journal of Human–Computer Interaction*, 2018.
[5] K. Rahman, Science of Selenium: Master Web UI Automation and Create Your Own Test Automation Framework, Noida: BPB Publications, 2019.
[6] A. AxelRod, (2018). Complete Guide to Test Automation. Matan: Apress
[7] S. v. Broucke dan B. Baesens, Practical Web Scraping for Data Science, Leuven: Apress, 2018.
[8] R. U. Rahman dan D. S. Tomar, "Threats of price scraping on e-commerce websites: attack model and its detection using neural network," *Journal of Computer Virology and Hacking Techniques*, 2019.
[9] E. Uzun, "A Novel Web Scraping Approach Using the Additional Information Obtained From Web Pages," *IEEE Access*, 2020.

[10] R. DIOUF, E. N. SARR, O. SALL, B. BIRREGAH, M. BOUSSO dan S. N. MBAYE, "Web Scraping: State-of-the-Art and Areas of Application," *IEEE International Conference on Big Data (Big Data)*, 2019.
[11] T. W. Miller, Web and Network Data Science: Modeling Techniques in Predictive Analytics, Bergen County: Pearson Education, Inc., 2015.
[12] U. Gundecha, Instant Selenium Testing Tools Starter, Birmingham: Packt Publishing , 2013.
[13] U. Gundecha dan S. Avasarala, Selenium WebDriver 3 Practical Guide, Birmingham: Packt Publisher, 2018.
[14] TestNG," [Online]. Available: https://testng.org/doc/. [Diakses 7 June 2021]
[15] V. Menon, TestNG Beginner's Guide, Birmingham: Packt Publishing, 2013